CLEAN COPY OF AMENDED PARAGRAPH [0029]

[0029] Continuing with the discussion of Figure 2, each block of dynamic memory (112, 114) is distributed across the processors (*i.e.,* Processor A (116) and Processor B (118)) within the system. In this case, each block of dynamic memory (112, 114) is distributed across two processors (116, 118) resulting in a Consumer 1 Processor A ("C1 PA") dynamic memory block (120), a Consumer 1 Processor B ("C1 PB") dynamic memory block (122), a Consumer 2 Processor A ("C2 PA") dynamic memory block (124), and a Consumer 2 Processor B ("C2 PB") dynamic memory block (126). In addition, each dynamic memory block on each processor (*i.e.,*120, 122, 124, 126) is associated, via the tracing framework (100), with a series of lists (132, 134, 136, 138) defining the state of the various data chunks (defined below) within each per-consumer memory block (*i.e.,* 120, 122, 124, 126).

CLEAN COPY OF AMENDED PARAGRAPH [0037]

[0037] Figure 6 shows a flowchart in accordance with one embodiment of the invention. More specifically, Figure 6 shows a method of allocating a dynamic variable in accordance with one embodiment of the invention. Initially, a request to allocate a dynamic variable, typically in the form of a dynamic load or dynamic store operation, is received (Step 200). The tracing framework then determines if the dynamic variable has been previously allocated (Step 202). If the dynamic variable has been previously allocated, then the method ends. Otherwise, the tracing framework determines if the consumer dynamic memory state is FREE or CLEAN (Step 204). Note that the tracing framework includes functionality to determine the consumer and consumer dynamic memory state associated with the probe. If the consumer dynamic variable state is not CLEAN or FREE (Step 204), then the appropriate counters are incremented to indicate that a dynamic drop has occurred (*i.e.*, the dynamic variable could not be dynamically allocated because there are no data chunks available to allocate) (Step 206). Note the dynamic drop(s) may be defined with respect to the value of the consumer dynamic memory state when the drop occurred (*e.g.*, a dirty drop counter is incremented if a dynamic drop occurs when the consumer dynamic memory state is DIRTY). Once the appropriate counters have been updated the process ends.

CLEAN COPY OF AMENDED PARAGRAPH [0045]

[0045] In one or more embodiments of the invention, an asynchronous cleaner is used to "clean" the dirty data chunks, thereby making them available for reallocation. The cleaner performs a series of steps to ensure that, once the data chunks have been cleaned, all processors have completed manipulating the dynamic variable, and the data chunks in which the dynamic variable resides are free to be reallocated to a new dynamic variable.